MODELING THE PULTEC EQP-1A WITH WAVE DIGITAL FILTERS

Alberto BARRERA[1], **Xavier LIZARRAGA-SEIJAS**[1], and **Frederic FONT**[1]

[1]**Universitat Pompeu Fabra**, Barcelona, Spain

ABSTRACT

This paper details the development of a virtual analog model of the passive equalizer stage of the classic Pultec EQP-1A studio equalizer using Wave Digital Filters. The modeling process involves analyzing the unit's schematics, LTspice simulations, and implementing a Wave Digital Filter structure using the `pywdf` library in Python. The proposed structure initially used *R*-Type adaptors, that are removed in a second model, better optimized for real-time. The Python prototype is compared to LTspice simulations, showing that, at sufficiently high sampling rates, the discrepancies between the two are marginal. The Wave Digital Filter model was then ported to C++ using the `chowdsp_wdf` library. Both implementations of the model in C++ (the one that uses *R*-Type adaptors and the real-time optimized one) are tested. Our optimized implementation proved to be much faster than the other open-source, Faust-based emulation of the EQP1-A that we evaluated, and a more accurate emulation of the original unit's equalizer stage.

1. INTRODUCTION

1.1 Motivation

Analog audio gear has been used in music production for almost a century and its distinct character continues to be sought after by many producers and sound engineers. Analog devices, including compressors, equalizers, and amplifiers, are highly regarded for their ability to introduce rich harmonic content, saturation and nonlinear characteristics into the processed signal.

Analog gear, however, can be fragile, needs maintenance, takes up physical space in the studio and is more expensive to acquire than its digital counterparts. It is also less convenient to use since it does not allow to recall configurations or use presets (like software does). During the last three decades, the processes involved in music recording, editing and post-processing have shifted to the digital domain and an important part of the audio equipment industry has disappeared, making these devices more and more scarce. Addressing these problems, virtual analog modeling techniques have long been of great interest to replicate the sound characteristics of analog gear in the digital environment [1–3].

Virtual analog modeling methods can be categorized as black-box or white-box (or a mix of both, often referred to as grey-box). Black-box techniques consider the system as a whole while white-box approaches aim to model the system by focusing on the characteristics of its individual components and their interactions.

Black-box methods are usually better for capturing "unpredicted" behavior that is not reflected in the circuit schematic or where there is not a complete understanding of the mechanisms involved in the processing. As a disadvantage, control parameters do not always map to the model's parameters in obvious ways and a complex mapping is often needed. Some of the most common black-box techniques include swept-sine methods [4–6], Volterra Series [7, 8] or in the last few years, the use of Machine Learning and, specially, Deep Neural Networks [9–11].

White-box modeling, on the other hand, allow to accurately represent the system's behaviour from schematics even without having access to a real unit and allow for direct mapping of control parameters to specific components or variables of the model [12]. The development of these models require deep domain knowledge of physics, circuit theory and digital signal processing. The most common white-box modeling methods are Modified Nodal Analysis [13, 14] and Wave Digital Filters (WDF) [15]. While Modified Nodal Analysis remains extensively used for virtual analog modeling, the WDF approach has gained considerable recognition due to its simplicity, modularity and ease to adapt to real-time applications. These advantages, combined with the availability of software libraries for development and rapid prototyping of WDFs, makes it a very attractive method for modeling analog circuits [16, 17].

1.2 Wave Digital Filters

WDFs are a kind of digital filter based on physical modeling principles [15, 18]. They can be used to represent the physical state of an element, for example, the current physical state of a capacitor or an inductor. WDFs are also a kind of finite difference scheme with unusually good numerical properties.

In the WDF domain, each circuit element can be defined by a port with an incident and reflected wave and a free variable, the port resistance [15, 19]. The incident and reflected waves, a and b respectively, at a certain port are defined by equations (1) and (2). The port resistance, R_p can be set to any real value.

$$a = v(t) + R_p \cdot i(t) \tag{1}$$

$$b = v(t) - R_p \cdot i(t) \tag{2}$$

Where $v(t)$ and $i(t)$ are Kirchhoff's domain variables, the instantaneous voltage and current across and through the terminals of an element. Equations (3) and (4) are used to get $v(t)$ and $i(t)$ back from the WDF domain variables.

$$v(t) = \frac{a+b}{2} \tag{3}$$

$$i(t) = \frac{a-b}{2R_p} \tag{4}$$

In order to make the WDF structure, first it is necessary to discretize each of the components of the circuit. The usual method for this digitization is the bi-linear transform but other methods can be used. Table 1 details the resulting expressions for both the port impedance and reflected wave of common circuit elements

Table 1: Expression for the WDF's domain variables of the different circuit elements.

	$\boldsymbol{R_p}$	$\boldsymbol{b}$
Resistor	R	0
Capacitor	$1/(2f_sC)$	$z^{-1}a$
Inductor	$2f_sL$	$-z^{-1}a$
Voltage Source	Matches the rest of the circuit	$2v_s - a$

Where z^{-1} is a one sample delay and f_s is the sampling frequency.

To prevent instability due to delay-free loops when deriving the WDF elements, the reflected wave, $b[n]$, cannot be directly proportional to the instantaneous value of the incident wave, $a[n]$. However, it can be proportional to a delayed version of the incident wave, $a[n-\tau]$. Usually, choosing an appropriate value for R_p eliminates the possibility of such loops, as shown in [15].

Once all the components of the circuit have been individually discretized they can be connected using adaptors. Adaptors are two-port elements characterized by a reflection coefficient that determines what amount of each of the incident waves gets reflected into the connected components.

In traditional WDF theory it is only possible to connect elements using series or parallel adaptors; this limits the topologies that can be modeled to circuits in which all of the components are connected either in series or in parallel. The reflected waves to each of the elements connected to a WDF adaptor, b_1 and b_2, follow the expressions of table 2.

Table 2: Definition of the reflected waves for the WDF adaptors.

	$\boldsymbol{b_1}$	$\boldsymbol{b_2}$
Series	$-a_2 + \gamma \cdot (a_1 + a_2)$	$-a_1 + \gamma \cdot (a_1 + a_2)$
Parallel	$a_2 + \gamma \cdot (a_1 - a_2)$	$a_1 + \gamma \cdot (a_1 - a_2)$

Where γ is the reflection coefficient and a_1 and a_2 are the incident waves arriving from each element connected to the adaptor.

With all the circuit elements and adaptors defined, they are connected together in a SPQR tree structure [20], where all of the *Series*, *Parallel*, *Q* and *R* nodes are defined. At each computational cycle of a WDF algorithm it is necessary to first, recalculate and propagate the reflected waves towards the root of the tree. Then, the resulting incident wave is used to calculate the reflected wave at the root of the tree, which is then propagated downwards. Finally, the voltage values are calculated at the output of the circuit and values in reactive components are stored to be used in the next computational cycle.

Apart from the series and parallel topology limitation mentioned before, only one nonlinear element can be introduced into the circuit. To do so, the nonlinear element is placed at the root of the SPQR tree and all the branches are connected to it. To model circuits with complex topologies or circuits containing multiple nonlinear elements it is necessary to introduce special adaptors such as *N*-port or *R*-type adaptors [21], characterized by a scattering matrix that acts as a multidimensional reflection coefficient between all of the elements connected to the adaptor.

1.3 The Pultec EQP-1A Program Equalizer

The Pultec EQP-1A is a classic analog equalizer originally released in 1951 (as the EQP-1), becoming the first program equalizer to be introduced to the market. It was mainly used in radio stations to equalize the broadcast program. The unit has ever since become a favorite among audio professionals and has been used on innumerable recordings for mixing and mastering. It has also been cloned by different manufacturers and emulated as a digital processor by multiple audio plug-in companies.

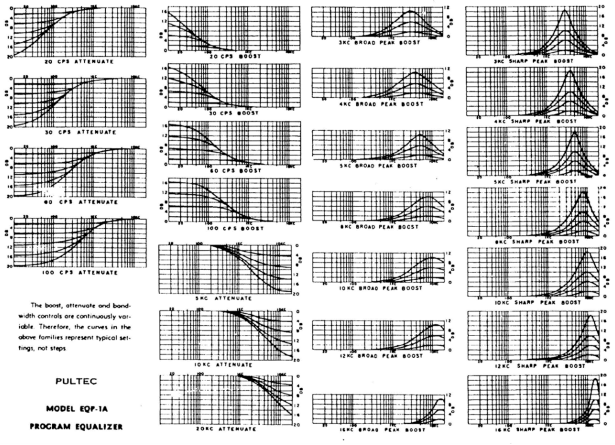


Figure 1: Frequency response curves from the EQP-1A user manual.

The EQP-1A consists of a passive equalizer stage and a vacuum tube gain stage to restore the insertion loss caused by the equalizer stage. The equalizer stage is composed by four different filters:

- Low frequency boost: shelving filter with selectable cutoff frequency and up to 16 dB of gain.
- Low frequency cut: shelving filter with up to 20 dB of attenuation.

- High frequency boost: resonant RLC filter with configurable center frequency and bandwith.
- High frequency cut: shelving filter with configurable cutoff frequency and up to 20 dB of attenuation.

The frequency response of each of the filters in different gain and frequency configurations from the original user manual are included in figure 1.

Although the EQP-1A's manual advises against boosting and attenuating simultaneously on the low frequencies, when both are applied at the same time, the boost and cut curves affect slightly different frequency bands. This allows to boost the low end, adding body to the sound while attenuating the low-mid frequency range to avoid "muddiness".

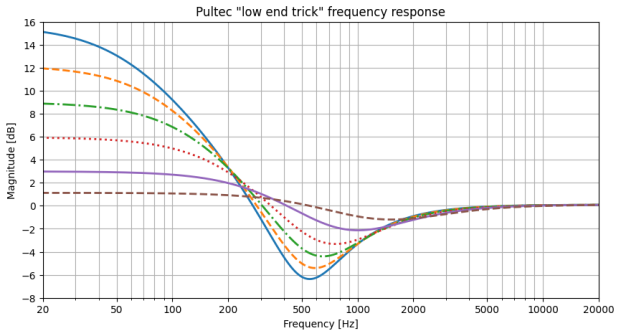


Figure 2: Frequency response of the EQP-1A when boosting and cutting the low frequency band.

Figure 2 illustrates the magnitude response of the equalizer when both boosting and attenuation are simultaneously applied at 60 Hz, with the boost and cut potentiometers set to the same position. This effect is commonly known as the "Pultec low end trick" and it is used extensively to enhance the sound of vocals, drums, guitars, or bass.

Because the equalizer stage is passive, it introduces signal loss effect. To make up for the 16 dB insertion loss, the signal level is restored using an output gain stage based on a 12AX7 and a 12AU7 vacuum tubes. Additionally, the original circuit incorporates input and output transformers. Nonetheless, this paper is focused on simulating the passive equalizer stage, with the acknowledgment that modeling the gain stage and input and output transformers of the unit is beyond its scope.

2. METHODOLOGY

2.1 References for the circuit

Since the original schematics for the circuit of the EQP-1A are not publicly available and we did not have access to the hardware unit, we took as reference a few documents that can be found online:

- The circuit diagram for the EQP-1R [22], another variant of the EQP-1A also built by Pulse Techniques.
- The open-source circuit diagram for the EQP-1A clone by Gyraf Audio [23].
- The frequency response curves from the EQP-1A user manual [24], already shown in figure 1.
- The frequency response curves from the EQP-1A 2019 reissue made by the original manufacturer taken from an article published in the Sound on Sound magazine [25] and measured using laboratory equipment.

Considering all of them we can gain a good understanding about the topology and components of the original device as well as its frequency response.

2.2 Circuit Simulations

Based on the references for the circuit, a first version of the circuit was simulated in LTspice. This first simulation showed some discrepancies with the measurements taken from the 2019 reissue of the unit so the value of some components was adjusted to closely match them.

Since the model does not need to adapt the input and output impedance of the circuit, the input and output resistors are removed, adjusting the necessary components to maintain the circuit's behavior. This would allow us to model the circuit in the WDF domain without the need of *R*-Type adaptors, making the processing faster for the same circuit's response. The version of the circuit used for the final model is included in figure 3.

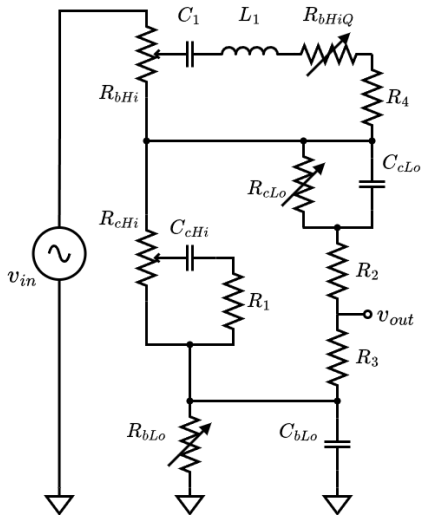
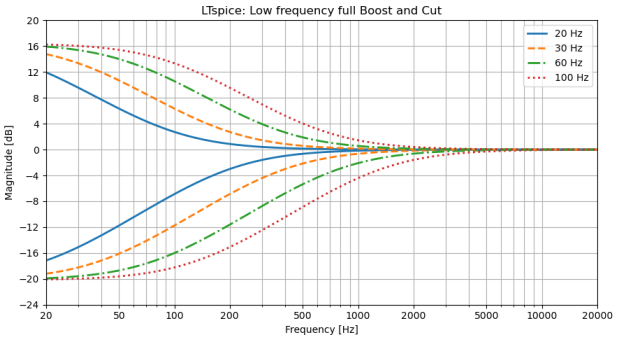


Figure 3: Modified version of the circuit used for the final model.

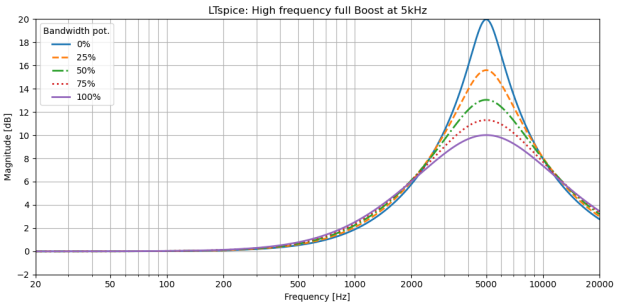
The results of the simulations carried out on this modified version of the circuit are included for: the maximum low frequency boost and cut configurations for all of the selectable frequencies (figure 4a), different bandwidth values for the resonant high frequency filter at maximum boost at 5 kHz (figure 4b) and maximum cut for all of the selectable frequencies of the high shelving filter (figure 4c).

All of the simulations were carried out with the same configuration as the measurements taken from the hardware

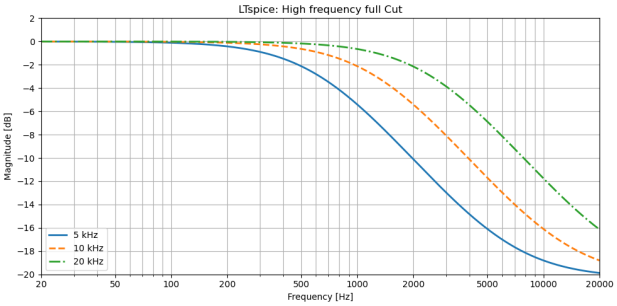
unit in the Sound and Sound article [25] so they could be directly compared against them. Although they could not be compared numerically in a meaningful way, the achieved response curves are identical.



(a) Low frequency boost and cut



(b) High frequency boost at 5kHz



(c) High frequency cut

Figure 4: LTspice simulations frequency responses

2.3 Wave Digital Filter Implementation

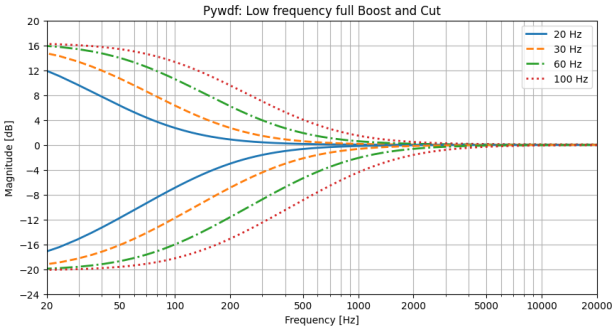
Once the frequency response of the simulations closely matches the references, the circuit is constructed in the WDF domain using Python and the `pywdf` library [1] [16]. `pywdf` includes WDF models for resistors, capacitors and inductors, as well as series, parallel and *R*-Type adaptors.

A first implementation of the circuit, that used *R*-Type adaptors, was developed following the circuit of the references. To derive the *R*-Type adaptor, the circuit's nodes and ports are labeled to create a net-list that describes the circuit. With this net-list the scattering matrix can be computed using *R*-Solver and SageMath. The process of obtaining the scattering matrix from the circuit is explained in an article published online by Jatin Chowdhury [26].

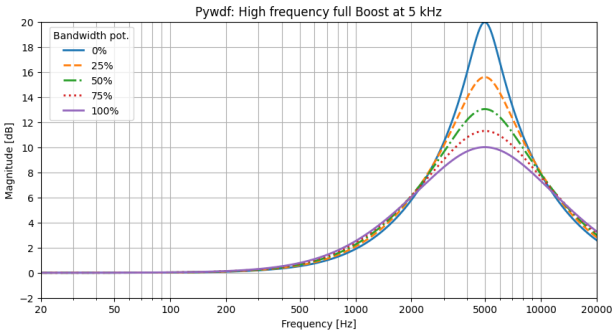
This first model, that uses *R*-Type adaptors, had the same response as the final version but worse performance. The performance of both models is compared in section 3.2.

The final WDF implementation of the circuit followed the circuit on figure 3, which resulted in the WDF structure represented in figure 6. The voltage source is connected at the root of the circuit and all of the components are connected through series and parallel adaptors to it.

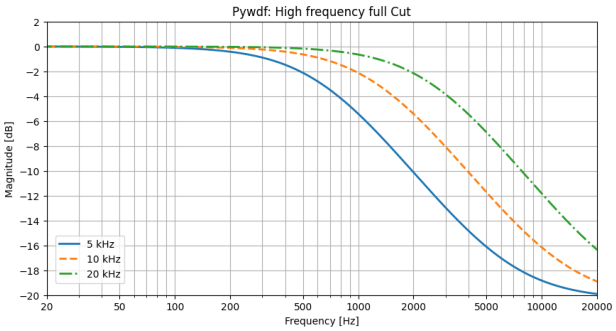
Using the `pywdf` model and the functions included in the library we can easily obtain the frequency response of the circuit. The frequency responses of the model for the same configurations as in the simulations carried out in LTspice and the ones measured from the hardware unit were computed at a sampling rate of 192 kHz and are included in figures 5a, 5b and 5c.



(a) Low frequency boost and cut



(b) High frequency boost at 5kHz



(c) High frequency cut

Figure 5: WDF model frequency responses

The frequency responses obtained are very accurate at

[1] `https://github.com/gusanthon/pywdf`

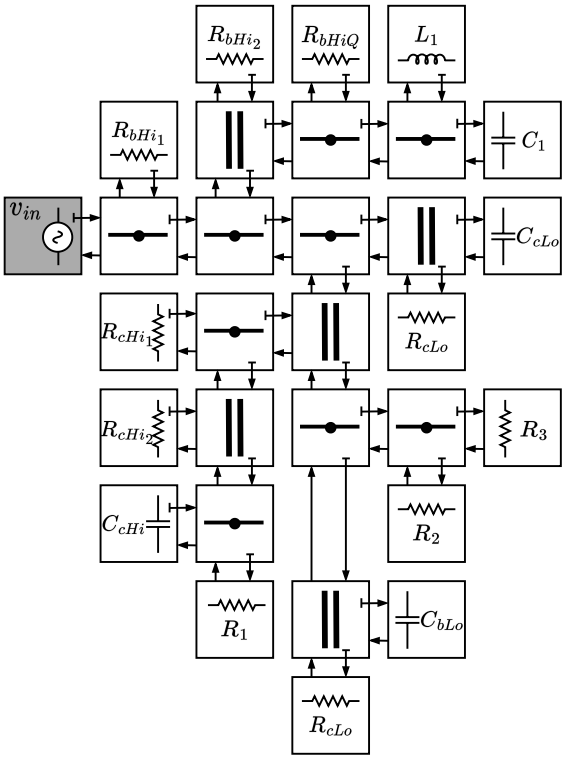


Figure 6: WDF diagram of the passive equalizer stage.

high sampling rates, but the Python models are slow and only able to process audio offline. While Python and `pywdf` provides a valuable prototyping environment, its limitations become apparent in real-time applications where faster programming languages, like C++, are needed.

2.4 Real-time C++ Model

After achieving accurate results from the WDF model in Python, it is necessary to translate the WDF model to C++ to enable real-time operation. To achieve this we used the `chowdsp_wdf` library [2] [17] for the WDF modeling and the JUCE framework [3] to compile the model into an audio plug-in.

One big advantage of using `pywdf` library for prototyping in Python alongside the `chowdsp_wdf` library for the real-time implementation is that they share the same foundation: all the processes are equivalent and numerically they shed the same results. This ensures that a model prototyped in Python can be seamlessly ported to C++ without any alterations, preserving identically the model's topology and behavior.

At lower sampling rates, and due to the use of the bilinear transform during the discretization of the circuit elements, considerable cramping occurs on the higher end of the spectrum. For this reason, getting good accuracy out of the WDF model (as quantified in section 3.1) requires the use of high sampling rates above the conventional 48 kHz used in audio production. However, running an entire project at high sampling rates is not always efficient.

[2] `https://github.com/Chowdhury-DSP/chowdsp_wdf`
[3] `https://juce.com`

To address this, our plug-in performs a series of operations: it internally up-samples the input signal, processes the over-sampled signal using the WDF model, and then down-samples the processed signal back to the original host's sampling frequency. The plug-in provides options for x2, x4, or no oversampling.

The result of this process is a fully operational audio plug-in that effectively incorporates the WDF model within a real-time environment. The plug-in can be used in any Digital Audio Workstation and it is available to download on GitHub [4]. A snapshot of its GUI is included on figure 7.

Figure 7: Graphical user interface of the plugin.

3. RESULTS

3.1 Frequency Response

To asses the accuracy of our WDF implementation, the frequency response curves computed from the Python model are compared against the ones obtained through LTspice simulations and the error between both is calculated.

For each graph, both the frequency response obtained from LTspice and the one computed from `pywdf` are displayed. The background color indicates the squared error between the two, $(|H_{\texttt{LTspice}}(f)| - |H_{\texttt{pywdf}}(f)|)^2$, and the $RMSE$, that follows the definition of equation (5), is included at the bottom left corner.

$$RMSE = \sqrt{\frac{1}{N}\sum_{k}(|H_{\texttt{spice}}(k\Delta f)| - |H_{\texttt{wdf}}(k\Delta f)|)^2} \quad (5)$$

The error graphs for an example EQP-1A configuration are included in figures 8, 9 and 10 for sampling frequencies of 48 kHz, 96 kHz and 192 kHz respectively. The numerical values for the error of the individual filters at the the most adverse configurations are gathered in table 3.

Table 3: Magnitude response error on each filter at the most adverse configuration.

	LF Boost	**LF Cut**	**HF Boost**	**HF Cut**
fs (kHz)	**RMSE**	**RMSE**	**RMSE**	**RMSE**
48	3.65e-2	2.07e-3	1.40	3.60e-2
96	1.86e-2	5.43e-4	6.44e-1	1.28e-2
192	9.37e-3	1.39e-4	2.20e-1	3.96e-3

[4] `https://github.com/ABSounds/EQP-WDF-1A`

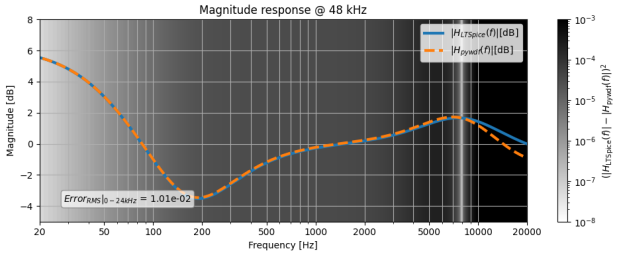


Figure 8: Example configuration frequency response error at 48 kHz.

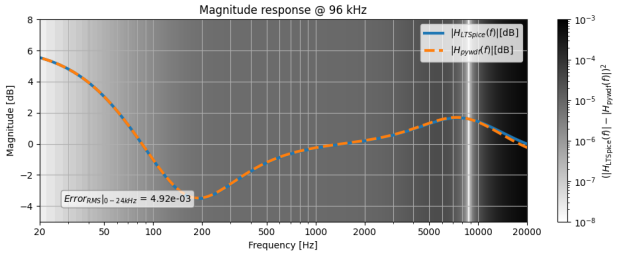


Figure 9: Example configuration frequency response error at 96 kHz.

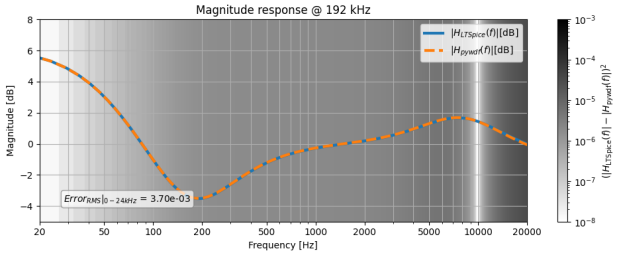


Figure 10: Example configuration frequency response error at 192 kHz.

The comparisons show that at higher sampling rates the model is very accurate and the error between the simulations and the WDF model are minimal.

Although we focused on the model's magnitude response in this work, we also examined its phase response and found no significant discrepancies from the expected behavior of the unit. The phase response does exhibit distortion when approaching the Nyquist frequency; however, as with the magnitude response error, this distortion can be reduced in the audio range by processing at higher sample rates.

3.2 Performance

To assess the performance of the different models, a random audio spanning 20 seconds is generated and processed using each model. The time taken to process all samples is measured using Python's `time.monotonic()` function from the standard library `time`.

The Python implementations are executed directly on a Jupyter Notebook, while the VST3 plug-ins are loaded into Python using Spotify's Pedalboard library [5] . All the processing is executed locally on a computer with an i7-8550U processor running at 1.8 GHz and 16 GB of RAM operating on Windows 11.

The evaluation includes the processing time for the 20 seconds of audio, as well as a real-time ratio. This ratio indicates how many times faster than real-time the model can operate. For instance, a ratio of 2 implies that the model is able to process 2 seconds of audio in 1 second. Any ratio below 1 means that the model is not suitable for real-time use.

Table 4 includes the comparison between the performance of our two Python models developed using the `pywdf` library. The first one uses traditional WDF with the modified version of the circuit, while the other one uses *R*-Type adaptors.

Table 4: Time taken to process 20 s of audio with each of the Python models.

	pywdf		**pywdf w/ R-Type**	
fs (kHz)	**Time (s)**	**Ratio**	**Time (s)**	**Ratio**
48	**26.969**	**0.74**	120.775	0.17
96	**53.931**	**0.37**	229.953	0.09
192	**108.984**	**0.18**	451.212	0.04

The results show that the implementation that used *R*-Type adaptors was 75% slower than the final one. Working at typical audio rates (48, 96 and 192 kHz), none of them was able to run in real-time. This was expected and caused by the nature of the Python programming language.

Table 5 includes the performance assessment of our C++ model compiled into a VST3 plug-in (EQP-WDF-1A) and compares it to another implementation of the EQP-1A described on this paper [27] developed in the Faust programming language that uses *R*-Type adaptors and was also compiled to VST3.

Table 5: Time taken to process 20 s of audio with each of the VST3 plug-ins.

	EQP-WDF-1A		**Faust EQP-1A (R-Type)**	
fs (kHz)	**Time (s)**	**Ratio**	**Time (s)**	**Ratio**
48	**0.092**	**217**	0.175	114
96	**0.182**	**110**	0.326	61
192	**0.360**	**56**	0.638	31

Overall, our plug-in performed between 40% to 47% faster than the Faust implementation on every configuration.

[5] `https://github.com/spotify/pedalboard`

4. CONCLUSIONS

Based on our measurements we can conclude that our WDF model accurately replicates the frequency response of the Pultec EQP-1A, specially at higher sampling rates, as confirmed by the comparisons between the LTspice simulations, the Python WDF model frequency response and measurements taken from the 2019 reissue of the unit.

The C++ implementation of the WDF model is also efficient and capable of real-time operation. It performs faster than existing implementations of the circuit and yields better results. Thanks to the use of the `pywdf` and `chowdsp_wdf` libraries, transitioning from the Python prototype to a real-time C++ implementation is a smooth process that preserves the model behaviour intact with some minor adaptations to the Python code.

This development was a first step in the direction to a complete open-source model of the Pultec EQP-1A. Although the frequency response achieved is faithful to the original unit, other sections of the circuit (specially the gain stage and input/output transformers) would need to be modeled to recreate its harmonic properties.

4.1 Future work

Various methods can be used to model the distortion produced by the 12AX7 and 12AU7 vacuum tubes in the EQP-1A's gain stage.

One option would be to use a filtering block and a static waveshaper as proposed in [28, 29]. This is a simple to implement but won't fully model the non-symmetrical distortion present in circuits with pairs of triodes like this one.

A more accurate option would involve integrating vacuum tube models into a WDF implementation, as discussed in [30, 31]. This would offer a high accuracy but might prove challenging due to the complex interaction between the gain and equalizer stages.

Instead of the WDF models of the vacuum tubes, neural networks can be introduced into the WDF domain model as proposed in [32]. This has shown great results in modeling guitar amplifiers and would likely work as well for the gain stage of the EQP-1A but it would not reduce the complexity of the WDF model.

Acknowledgments

This research was carried out under the project "IA y Música: Cátedra en Inteligencia Artificial y Música" (TSI-100929-2023-1), financed by Secretaría de Estado de Digitalización e Inteligencia Artificial, and Unión Europea-Next Generation EU, under the program Cátedras ENIA 2022. This paper is also based on the research conducted during a Master's Thesis on Sound and Music Computing at the Universitat Pompeu Fabra in Barcelona. Our gratitude goes to Audio Merge's[6] team for leading this proposal and to Xavier Lizarraga and Frederic Font, members of the Music Technology Group at UPF, for guiding. Their support and expertise were key to the success of this research.

[6] `https://audio-merge.com`

5. REFERENCES

[1] S. D'Angelo, "Lightweight virtual analog modeling," in *Proceedings of the 22nd Colloquium on Music Informatics, Udine, Italy*. Independent Researcher, 2018, pp. 20–23.

[2] A. Bernardini, A. Sarti *et al.*, "Towards inverse virtual analog modeling," in *Proc. 22nd International Conference on Digital Audio Effects (DAFx 2019)*, 2019, pp. 1–8.

[3] V. Valimaki, J. D. Parker, L. Savioja, J. O. Smith, and J. S. Abel, "Fifty years of artificial reverberation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 5, pp. 1421–1448, 2012.

[4] A. J. Berkhout, D. de Vries, and M. M. Boone, "A new method to acquire impulse responses in concert halls," *The Journal of the Acoustical Society of America*, vol. 68, no. 1, pp. 179–183, 1980.

[5] A. Farina, A. Bellini, E. Armelloni *et al.*, "Non-linear convolution: A new approach for the auralization of distorting systems," *Preprints-Audio Engineering Society*, 2001.

[6] A. Farina, "Simultaneous measurement of impulse response and distortion with a swept-sine technique," in *Audio engineering society convention 108*. Audio Engineering Society, 2000.

[7] T. Hélie, "Volterra series and state transformation for real-time simulations of audio circuits including saturations: Application to the moog ladder filter," *IEEE Transactions on Audio, Speech Language Processing*, vol. 18, pp. 747–759, 2010.

[8] A. Farina, E. Armelloni *et al.*, "Emulation of not-linear, time-variant devices by the convolution technique," in *Congresso AES Italia, Como*, 2005.

[9] J. Chowdhury, "A Comparison of Virtual Analog Modelling Techniques for Desktop and Embedded Implementations," Sep. 2020, arXiv:2009.02833 [cs, eess]. [Online]. Available: http://arxiv.org/abs/2009.02833

[10] A. Wright, E.-P. Damskägg, and V. Välimäki, "Real-Time Black-Box Modelling with Recurrent Neural Networks," in *Proceedings of the International Conference on Digital Audio Effects*. United Kingdom: University of Birmingham, Sep. 2019.

[11] C. J. Steinmetz, J. Pons, S. Pascual, and J. Serrà, "Automatic multitrack mixing with a differentiable mixing console of neural audio effects," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 71–75.

[12] M. Verasani, A. Bernardini, and A. Sarti, "Modeling sallen-key audio filters in the wave digital domain," in *2017 IEEE international conference on acoustics,*

speech and signal processing (ICASSP). IEEE, 2017, pp. 431–435.

[13] C.-W. Ho, A. Ruehli, and P. Brennan, "The modified nodal approach to network analysis," *IEEE Transactions on Circuits and Systems*, vol. 22, no. 6, pp. 504–509, Jun. 1975.

[14] K. Dempwolf, M. Holters, and U. Zölzer, "Discretization of parametric analog circuits for real-time simulations," in *Proceedings of the 13th international conference on digital audio effects (DAFx'10)*, 2010.

[15] A. Fettweis, "Wave digital filters: Theory and practice," *Proceedings of the IEEE*, vol. 74, no. 2, pp. 270–327, Feb. 1986.

[16] G. Anthon, X. Lizarraga-Seijas, and F. Font, "Pywdf: an open source library for prototyping and simulating wave filter circuits in python," in *Proceedings of the 26th International Conference on Digital Audio Effects (DAFx-23)*, Denmark, 2023, pp. 335–341.

[17] J.Chowdhury, "chowdsp_wdf: An Advanced C++ Library for Wave Digital Circuit Modelling," Oct. 2022, arXiv:2210.12554 [eess]. [Online]. Available: http://arxiv.org/abs/2210.12554

[18] J. O. Smith, *Physical Audio Signal Processing*, ch. Wave Digital Filters, online book, 2010 edition. [Online]. Available: https://ccrma.stanford.edu/~jos/pasp/Wave_Digital_Filters_I.html

[19] multivac61, "multivac61/wave_digital_notebook," May 2023, original-date: 2016-10-23. [Online]. Available: https://github.com/multivac61/wave_digital_notebook

[20] K. J. Werner, "Virtual Analog Modeling of Audio Circuitry Using Wave Digital Filters," Ph.D. dissertation, Stanford University, Stanford, 2016, pages: 261 Volume: Ph.D. in Computer-Based Music Theory and Acoustics.

[21] K. J. Werner, V. Nangia, J. O. Smith III, and J. S. Abel, "Resolving wave digital filters with multiple/multiport nonlinearities," in *Proc. 18th Conf. Digital Audio Effects*, 2015, pp. 387–394.

[22] "Dave Library." [Online]. Available: http://www.davegroupjapan.com/ekoukoku9.html

[23] Gyraf Audio, "The G-Pultec." [Online]. Available: https://www.gyraf.dk/gy_pd/pultec/pultec.htm

[24] Pulse Techniques, "Pultec EQP-1A User Manual." [Online]. Available: https://www.thehistoryofrecording.com/Manuals/Pultec/Pultec_EQP-1A_Manual.pdf

[25] H. Robjohns, "Pulse Techniques EQP-1A," *Sound on Sound*, Feb. 2019. [Online]. Available: https://www.soundonsound.com/reviews/pulse-techniques-eqp-1a

[26] J. Chowdhury, "Wave Digital Circuit Models with R-Type Adaptors," Oct. 2022. [Online]. Available: https://jatinchowdhury18.medium.com/wave-digital-filter-circuit-models-with-r-type-adaptors-39ad0ad658ce

[27] D. Roosenburg, E. Stine, R. Michon, and J. Chowdhury, "A wave digital filter modeling library for the faust programming language," in *Proc. 18th Sound Music Comput. Conf., Virtual*, 2021, pp. 24–30.

[28] R. Kuehnel, "Digital Modeling of Guitar Amplifier Preamp Distortion." [Online]. Available: https://www.ampbooks.com/mobile/dsp/preamp/

[29] I. Cohen, "Fifty shades of distortion," in *ADC'17*, London, UK, Nov. 2017. [Online]. Available: https://www.youtube.com/watch?v=oIChUOV_0w4

[30] M. Karjalainen and J. Pakarinen, "Wave digital simulation of a vacuum-tube amplifier," in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 5. IEEE, 2006.

[31] J. Pakarinen and M. Karjalainen, "Enhanced wave digital triode model for real-time tube amplifier emulation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 4, pp. 738–746, 2009.

[32] C. C. Darabundit, D. Roosenburg, and J. O. Smith III, "Neural net tube models for wave digital filters," in *Proc. 25th Int. Conf. Digital Audio Effects (DAFx20in22)*. Vienna University of Music and Performing Arts, 2022, pp. 153–160.